AMENDMENTS TO THE CLAIMS

Please cancel Claims 12-30 without prejudice.

1. (Previously Presented) A method for renaming memory references to stack locations in a computer processing system comprising a plurality of processors, comprising the steps of:

fetching, by a first processor, an instruction referencing a stack;

replacing the instruction referencing the stack with a references to a processor-internal registers of the first processor and entering the reference to the processor-internal register in a dispatch table upon determining that the instruction uses an architecturally defined stack access method; and

performing a consistency-preserving operation to recover an in-order value for the instruction referencing the stack from a main memory, and entering the in-order value in the dispatch table upon determining that the stack reference references a register of a second processor.

2. (Cancelled)

3. (Previously Presented) The program storage device according to claim 40, wherein said synchronizing step comprises the step of inserting in-order write operations for all of the stack references that are write stack references.

4. (Cancelled)

5. (Previously Presented) The method according to claim 1, wherein said step of performing the consistency-preserving operation comprises the step of bypassing a value from a given processor-internal register to a load operation that references a stack area and that does not use the architecturally defined stack access methods.

6. (Previously Presented) The method according to claim 1, further comprising the step of synchronizing an architected state between the processor-internal registers and a main memory of the computer processing system, and wherein said step of performing the consistency-preserving operation recovers the in-order value for the stack reference from the main memory, upon performing said synchronizing step.

7. (Previously Presented) The method according to claim 6, wherein the in-order value is written to the main memory by an in-order write operation inserted into an instruction stream containing an instruction corresponding to the instruction referencing the stack, when the instruction referencing the stack is a write stack reference.

8. (Original) The method according to claim 6, further comprising the step of writing the in-order value to the main memory in response to a load operation that does not use the architecturally defined stack access methods.

9. (Previously Presented) The method according to claim 1, wherein said step of performing a consistency-preserving operation comprises the steps of:

discarding all out-of-order state;

synchronizing an architected state between the processor-internal registers and the main memory of the computer processing system; and

restarting execution after a store operation has been performed that does not use the architecturally defined stack access methods.

10. (Original) The method according to claim 1, wherein the architecturally defined stack access methods comprise memory accesses that use at least one of a stack pointer, a frame pointer, and an argument pointer.

11. (Original) The method according to claim 1, wherein the architecturally defined stack access methods comprise push, pop, and other stack manipulation operations.

12-30. (Cancelled)

31. (Previously Presented) A program storage device readable by machine, tangibly embodying a program of instructions executable by the machine to perform methods steps for renaming memory references to stack locations in a computer processing system comprising a plurality of processors, the method steps comprising:

fetching, by a first processor, an instruction referencing a stack;

replacing the instruction referencing the stack with a references to a processor-internal registers of the first processor and entering the reference to the processor-internal register in a dispatch table upon determining that the instruction uses an architecturally defined stack access method; and

performing a consistency-preserving operation to recover an in-order value for the instruction referencing the stack from a main memory, and entering the in-order value in the dispatch table upon determining that the stack reference references a register of a second processor.

32. (Cancelled)

33. (Previously Presented) The program storage device according to claim 41, wherein said synchronizing step comprises the step of inserting in-order write operations for all of the stack references that are write stack references.

34. (Cancelled)

35. (Previously Presented) The program storage device according to claim 31, wherein said step of performing a consistency-preserving operation comprises the step of bypassing a value from a given processor-internal register to a load operation that references a stack area and that does not use the architecturally defined stack access methods.

36. (Previously Presented) The program storage device according to claim 31, further comprising the step of synchronizing an architected state between the processor-internal registers and a main memory of the computer processing system, and wherein said step of performing the consistency-preserving operation recovers the in-order value for the stack reference from the main memory, upon performing said synchronizing step.

37. (Previously Presented) The program storage device according to claim 36, wherein the in-order value is written to the main memory by an in-order write operation inserted into an instruction stream containing an instruction corresponding to the instruction referencing the stack, when the instruction referencing the stack is a write stack reference.

38. (Original) The program storage device according to claim 36, further comprising the step of writing the in-order value to the main memory in response to a load operation that does not use the architecturally defined stack access methods.

39. (Previously Presented) The program storage device according to claim 31, wherein said step of performing a consistency-preserving operation comprises the steps of:

discarding all out-of-order state;

synchronizing an architected state between the processor-internal registers and the main memory of the computer processing system; and

restarting execution after a store operation has been performed that does not use the architecturally defined stack access methods.

40. (Previously Presented) The method according to claim 1, further comprising synchronizing an architected state between the processor-internal registers and a main memory of the computer processing system.

41. (Previously Presented) The program storage device according to claim 31, further comprising synchronizing an architected state between the processor-internal registers and a main memory of the computer processing system.